SCALER School of Technology × Priyansh Saxena

CNN Architectures: From LeNet to ResNet

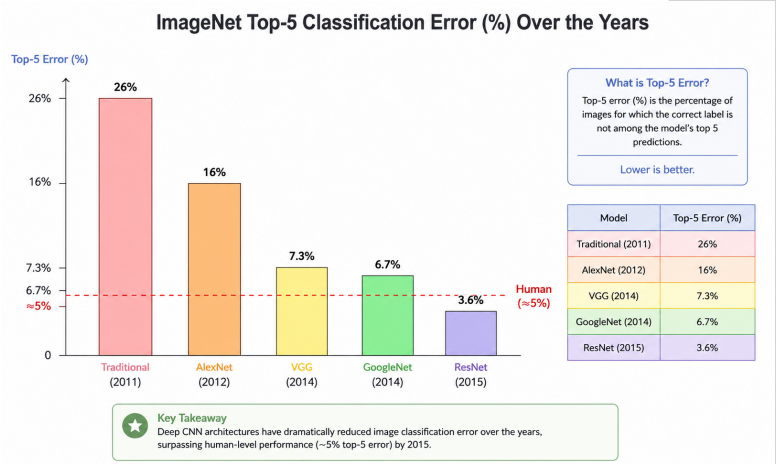
LeNet-5 • AlexNet • VGG • Inception • ResNet • Design Principles

Priyansh Saxena

Scaler School of Technology | Deep Learning I: Neural Networks | Week 4 — Session 2/2

2026

The ImageNet Timeline

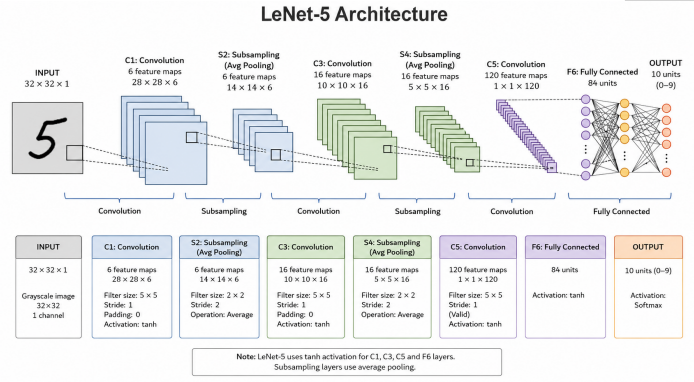


3 years: from barely competing to surpassing human accuracy.

Today’s Journey

Each architecture solved a **specific limitation** of its predecessor.

LeNet-5 Architecture (1998)



- Yann LeCun, 1998 — designed for handwritten digit recognition
- **60,000 parameters** total
- Used sigmoid/tanh activations
- Established the **Conv** → **Pool** → **Conv** → **Pool** → **FC** pattern

LeNet — Key Takeaways

The “Funnel” Pattern

Spatial dimensions **shrink**: $32 \to 28 \to 14 \to 10 \to 5$

Channel dimensions **grow**: $1 \to 6 \to 16$

- **Local connectivity** — each neuron sees only a small patch
- **Weight sharing** — same filter applied everywhere
- **Subsampling** — pooling reduces spatial size progressively

Limitation

Too small for natural images (224×224). Sigmoid/tanh cause vanishing gradients in deeper versions.

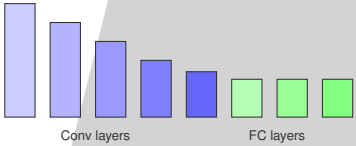
AlexNet — The 2012 Breakthrough

- Won ImageNet 2012: **16.4%** top-5 error
- Runner-up (non-DL): 26.2%
- 5 conv layers + 3 FC layers
- **60 million parameters**
- Split across 2 GPUs (necessity $\rightarrow$ innovation)

The "Big Bang" of Deep Learning

This single result convinced the world that neural networks actually work at scale.

Architecture sketch:



What AlexNet Changed

1. **ReLU activation** — 6× faster training than tanh
2. **Dropout** (0.5) in FC layers — regularization without extra data
3. **Data augmentation** — random crops, horizontal flips, color jitter
4. **GPU training** — made 60M parameters feasible
5. Local Response Normalization (now obsolete — replaced by BatchNorm)

The Lesson

It wasn't one trick — it was the **combination** of ReLU + Dropout + Augmentation + GPU that unlocked scale.

LeNet vs AlexNet — Scale Comparison

	LeNet-5 (1998)	AlexNet (2012)
Input size	32×32	224×224
Parameters	60K	60M
Depth	5 layers	8 layers
Activation	Sigmoid/Tanh	ReLU
Training	CPU, minutes	GPU, days
Task	10 digits	1000 classes

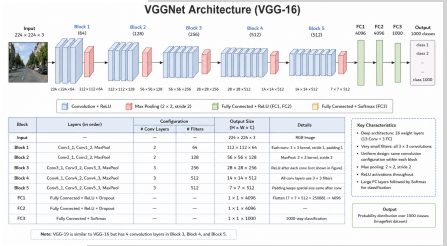
"Same recipe, 1000$\times$ bigger kitchen."

Quick Check

1. What single innovation had the biggest impact on AlexNet's **training speed**?
   - ✓ ReLU — no saturation, fast gradients
2. Why couldn't LeNet-5 be directly applied to ImageNet?
   - ✓ 60K params can't model 1000 classes on 224×224 images — too little capacity

VGGNet — The 3×3 Philosophy (2014)

- Oxford Visual Geometry Group
- Core idea: **only use 3×3 convolutions**
- VGG-16: 13 conv + 3 FC = 16 layers
- VGG-19: 16 conv + 3 FC = 19 layers
- **138M parameters** (mostly in FC layers)
- 7.3% top-5 error on ImageNet



Why 3×3 Works

Receptive Field Equivalence

- Two 3×3 convs = same receptive field as one 5×5
- Three 3×3 convs = same receptive field as one 7×7

One 7×7 filter

Parameters: $7 \times 7 \times C^2 = 49C^2$
Non-linearities: 1

Three 3×3 filters

Parameters: $3 \times (3 \times 3 \times C^2) = 27C^2$
Non-linearities: 3

45% fewer parameters AND **3× more non-linearity** → more expressive!

VGG Design Pattern

The Rule: Halve Spatial, Double Channels

$$224 \to 112 \to 56 \to 28 \to 14 \to 7 \quad \text{(spatial)}$$
$$64 \to 128 \to 256 \to 512 \to 512 \quad \text{(channels)}$$

- This keeps **computation per layer roughly constant**
- Simple, repeatable blocks — easy to scale
- Became the “reference architecture” for years

The Downside

138M parameters, with **124M in FC layers alone**. Very wasteful!

The Problem with Going Wider

Idea: Use multiple filter sizes (1×1, 3×3, 5×5) in parallel.

Problem: Computational explosion!

Example (Naive)

Input: $28 \times 28 \times 256$ channels

Apply 256 filters of each size in parallel:

- 256 filters of 1×1: $28 \times 28 \times 256$
- 256 filters of 3×3: $28 \times 28 \times 256$
- 256 filters of 5×5: $28 \times 28 \times 256$

Concatenated output: $28 \times 28 \times 768$ — and it keeps **growing**!

We need a way to control the channel explosion...

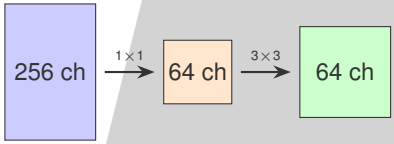
1×1 Convolutions — The Bottleneck Trick

What is a 1×1 conv?

- A fully-connected layer applied at **each pixel** independently
- Mixes channels without touching spatial dimensions
- Can **reduce** channels cheaply: 256 → 64

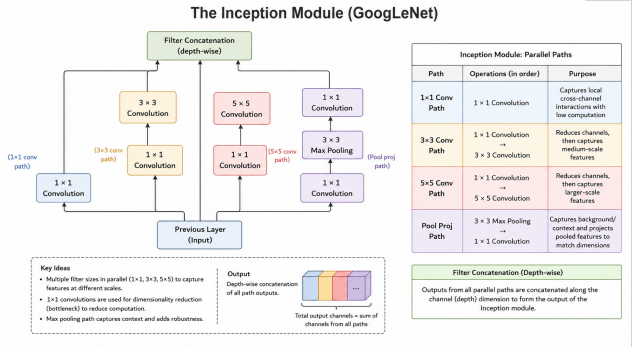
Why it helps:

- Reduce 256 channels to 64 with 1×1
- *Then* apply expensive 3×3 or 5×5 on 64 channels
- Massive computation savings!



Bottleneck: squeeze then process

The Inception Module



Inception Module: Parallel Paths

Path	Operations (in order)	Purpose
1×1 Conv Path	1 × 1 Convolution	Captures local cross-channel interactions with low computation
3×3 Conv Path	1 × 1 Convolution → 3 × 3 Convolution	Reduces channels, then captures medium-scale features
5×5 Conv Path	1 × 1 Convolution → 5 × 5 Convolution	Reduces channels, then captures larger-scale features
Pool Proj Path	3 × 3 Max Pooling → 1 × 1 Convolution	Captures background/ context and projects pooled features to match dimensions

Filter Concatenation (Depth-wise)

Outputs from all parallel paths are concatenated along the channel (depth) dimension to form the output of the Inception module.

“Don’t choose the filter size — use them ALL and let the network decide.”

GoogLeNet Results

- 22 layers deep
- Only **5M parameters** (vs VGG's 138M!)
- **No large FC layers** — uses Global Average Pooling
- 6.7% top-5 error — won ImageNet 2014
- 12× fewer params than AlexNet, **better accuracy**

Model	Params
AlexNet	60M
VGG-16	138M
GoogLeNet	**5M**

Smarter > Bigger

Key Innovation: Global Average Pooling

Instead of flattening into a huge FC layer, average each channel's spatial map into a single number. Eliminates millions of parameters.

Quick Check

1. What's the purpose of 1×1 convolutions in the Inception module?
   - ✓ Reduce channel dimensionality before expensive convolutions (bottleneck)

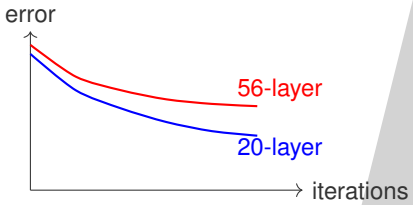
2. Which has more parameters: VGG-16 or GoogLeNet?
   - ✓ VGG-16 — by about $28\times$ (138M vs 5M)

The Degradation Problem

Surprising Result

A 56-layer plain network has **higher training error** than a 20-layer network!

- This is **NOT overfitting** — training error is worse, not just test error
- Deeper network can't even learn the identity ("do nothing") in extra layers
- It's an **optimization problem**: harder to train, not harder to generalize

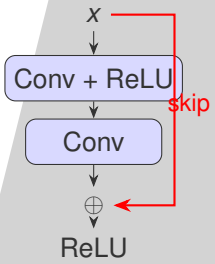


Residual Learning — The Key Insight

Instead of learning: $H(x)$ directly

Learn the residual: $F(x) = H(x) - x$

Output: $F(x) + x$



Why this helps

If the optimal mapping is close to identity:

- Learning $F(x) = 0$ is easy (push weights to zero)
- Learning $H(x) = x$ directly is hard!

Why Skip Connections Work

1. **Gradient highway** — gradients flow directly through the skip path
   - No multiplication chain $\rightarrow$ no vanishing!
2. **Easy identity baseline** — worst case, layer learns $F(x) = 0$
   - Output is just x — no harm done
3. **Ensemble effect** — ResNet behaves like an ensemble of many shallow paths

Analogy

"A highway with exit ramps. Traffic (gradients) can flow straight through even if some exits (layers) are blocked."

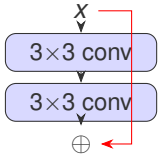
ResNet Results (2015)

- ResNet-152: **3.57% top-5 error** — superhuman!
- Won ImageNet, COCO detection, and COCO segmentation (all three!)
- Key insight: deeper **IS** better — when you have skip connections

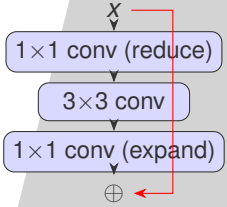
Variant	Layers	Params	Top-5 Error
ResNet-18	18	11M	10.9%
ResNet-34	34	21M	7.8%
ResNet-50	50	25M	6.7%
ResNet-101	101	44M	6.0%
ResNet-152	152	60M	3.6%

Residual Block Variants

Basic Block
(ResNet-18/34)



Bottleneck Block
(ResNet-50/101/152)



Bottleneck uses 1×1 convs (same idea as Inception!) to keep computation manageable in deep networks.

Architecture Design Rules

1. **Use small filters** (3×3) stacked deep — more non-linearity, fewer params
2. **Halve spatial, double channels** at each stage — keeps computation balanced
3. **Use 1×1 convs** for channel reduction (bottleneck)
4. **Add skip connections** when going beyond $\sim$20 layers
5. **Global Average Pooling** instead of large FC layers at the end

Architecture	Year	Depth	Params	Top-5 Err
LeNet-5	1998	5	60K	—
AlexNet	2012	8	60M	16.4%
VGG-16	2014	16	138M	7.3%
GoogLeNet	2014	22	5M	6.7%
ResNet-152	2015	152	60M	3.6%

Thank You!

Questions?

Priyansh Saxena

All lecture materials available at:

www.priyanshsaxena.com/sst-deep-learning-nn